Niko Torkkel

Interaktiivisen visualisoinnin toteutus Unreal-pelimoottorilla: case tykkisluuppi

Metropolia Ammattikorkeakoulu

Medianomi (AMK)

Viestinnän koulutusohjelma

Opinnäytetyö

22.4.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Niko Torkkel Interaktiivisen visualisoinnin toteutus Unreal-pelimoottorilla: case tykkisluuppi 31 sivua + 2 liitettä 22.4.2014
Tutkinto	Medianomi (AMK)
Koulutusohjelma	Viestinnän koulutusohjelma
Suuntautumisvaihtoehto	3D-animointi ja -visualisointi
Ohjaaja	Lehtori Kristian Simolin

Opinnäytetyössä toteutettiin reaaliaikainen 3D-havainnollistamismalli Suomenlinnassa rakennetusta tykkisluupista. Tavoitteena oli toteuttaa intuitiivinen tapa tarkastella tykkisluuppia tietokoneelta.

Toteutus aloitettiin suunnitelemalla applikaation tärkeimmät ominaisuudet kuten laivamallin esittely ja ohjelmassa navigoiminen. Alussa myös valmisteltiin, retopologioitiin ja teksturoitiin laivamalli valmiiksi.

Laivamalli vietiin Unreal Editoriin, jossa applikaation ominaisuuksia alettiin implementoimaan. Suurimmaksi osaksi ominaisuudet tehtiin noodipohjaisella scripting-ohjelmalla nimeltä Kismet. Samaan aikaan laivamallille valmistettiin materiaalit Unreal Editorin materiaalieditorissa.

Suurimpien ominaisuuksien implementoinnin jälkeen alkoi käyttöliittymän toteutus. Käyttöliittymä toteutettiin Flash-ohjelmalla ja yhdistettiin applikaatioon Unreal Editorissa. Käyttöliittymän implementoinnin jälkeen alkoi projektin hiominen, jotta saataisiin aikaiseksi ammattimainen lopputulos. Hiomiseen kuului teknisten bugien etsiminen ja korjaus, splash-animaation valmistaminen ja peli-ikonin teko.

Lopputuloksena saatiin aikaiseksi interaktiivinen visualisointi tykkisluupista. Näen onnistuneeni luomaan siitä hiotun ja käyttäjäystävällisen applikaation ja oppineeni projektin aikana paljon.

Avainsanat	Interaktiivinen visualisointi, Unreal moottori, Tykkisluuppi

Author Title Number of Pages Date	Niko Torkkel Creating an interactive visualisation with Unreal game engine: case gunboat 31 pages + 2 appendices 22.4.2014
Degree	Bachelor of arts
Degree Programme	Culture and creative industries
Specialisation option	3D animation and visualisation
Instructor	Kristian Simolin Lecturer

The purpose of the thesis was to create a realtime 3D visualisation of a gunboat built in Suomenlinna. Point for the visualisation was to be an intuitive way to see the gunboat from a computer.

The thesis was started by planning the most important aspects of the application like the representation of the gunboat and navigation inside the application. At the start of the project the gunboat was made ready, retopologized and textured.

After the model was ready it was imported to Unreal Editor where implementation of the most important features began. Most of the features were made with a node based scripting system called Kismet. At the same time materials were created for the gunboat in Unreal Editors own material editor.

When the main features were implemented it was time to start with the user interface. The user interface was created with Flash and then implemented to the application in Unreal Editor. After implementing the user interface it was time to start buffing up the application. This included hunting down bugs and for example creating the splash animation and game icon. This was necessary for the look and feel and to have a more professional feel for the application.

The end result was an interactive visualisation of the gunboat. I successed in creating a clean and user frendly application and had learned a lot through the project.

Keywords	Interactive visualisation, Unreal game engine, Gunboat

Sisällys

1 Johdanto

Tämän opinnäytetyön tarkoituksena oli toteuttaa reaaliaikainen 3D-havainnollistamismalli Suomenlinnassa rakennetusta tykkisluupista. Olin vuonna 2012 mallintamassa Suomenlinnassa silloin rakenteilla olevaa tykkisluuppia, ja halusin tehdä siitä myös käyttäjäystävällisen ohjelman, jossa laivasta kiinnostuneet ihmiset pääsisivät näkemään sen uusilla tavoilla.

Pääsin rakennusprojektin aikana seuraamaan laivan rakentamista paikan päällä, ja samoihin aikoihin alkoi muodostua idea reaaliaikaisesta applikaatiosta, jossa pääsisi tarkemmin tutustumaan tykkisluuppiin. Myöhemmin päätin tehdä aiheesta opinnäytetyöni. Päätöstä helpotti oma kasvava kiinnostus projektiin ja myös Metropolian 3D-animoinnin ja visualisoinnin yliopettajan Pasi Kaarron innostuneisuus tykkisluupista ja sen havainnollistamisesta.

Jo kesken tykkisuupin mallintamisen aloin hiomaan päässäni ideoita ja ominaisuuksia reaaliaikaiseen malliin. Myös mallintamisen aikana ilmenneet ongelmat vaikuttivat ideaan. Huomasin esimerkiksi, että laivan sisälle oli vaikea nähdä jo rakennusvaiheessa, saati sittenkun laiva on valmiiksi rakennettu ja runko suljettu.

Koska tein suurimman osan laivan mallintamisesta paikan päällä vanhassa levyhallissa, ajattelin samalla, että opinnäytetyöni olisi sopiva sijoitettavaksi juuri tykkisluupin lähelle joko pöytäkoneelle tai tabletille. Tabletilta toimiva versio olisi ollut ideaali tykkisluupin yleisöpurjehduksia varten, mutta en saanut sitä opinnäytetyöni rajoissa tehtyä. Toivottavasti kuitenkin ajattelemani käyttäjäystävällisyys näkyy lopputuloksessa.

Projekti oli samalla minulle mahtava tilaisuus syventää tietojani reaaliaikaisten moottorien kanssa työskentelystä. Minulla oli aikaisempaa kokemusta monista aiheen työvaiheista, mutta en ollut koskaan tehnyt alusta loppuun reaaliaikaista projektia itsekseni.

Tekstissäni käyn läpi projektin idean alkuperiä, kerron tekemistäni työvaiheista ja ongelmista mihin törmäsin. Pyrin myös kertomaan miten selvitin ongelmat aina kun mahdollista ja antamaan pieniä vinkkejä miten helpottaa työnkulkua.

Tavoitteenani oli siis toteuttaa intuitiivinen ja käyttäjäystävällinen 3D-havainnollistamismalli tykkisluupista, laajentaa tietämystäni reaaliaikamoottorien kanssa työskentelystä ja saada aikaiseksi ammattimainen ja hiottu lopputulos.

2 Keskeiset termit

UDK eli Unreal Developement Kit on paketti joka sisältää kaiken tarvittavan pelin tekemiseen Unreal moottorilla. Siihen kuuluu itse Unreal pelimoottori ja sen omat editorinsa.

Unreal Engine on itse pelimoottori jolla toteutin opinnäytetyöni.

Assetti (engl. Asset) tarkoittaa tämän opinnäytetyön yhteydessä Unreal Editorin ulkopuolella tehtyä asiaa, esimerkiksi tekstuuria tai 3D-mallia joka importoidaan editoriin.

Unreal Editor on Unreal moottorille tarkoitettu editori jossa yhdistetään kaikki muissa ohjelmissa tehdyt assetit ja jossa itse peli rakennetaan lopulliseen muotoonsa.

Noodi (engl. Node) on pala koodia visuaalisissa koodausohjelmissa.

Face on pinta, josta 3D-mallit koostuvat. Usein polygoni, mutta peligrafiikassa pinnat ovat kolmioituja, joten tämän opinnäytetyön yhteydessä tarkoitan termillä kolmikulmaista pintaa.

3 Suomenlinnan tykkisluuppiprojekti

Tykkisluupin ovat rakennuttaneet Ehrensvärd-seura ja Viaporin telakkasäätiö. Laiva on rakennettu Fredrik Henrik af Chapmannin 1700-luvun piirrustuksien mukaan, mutta myös niitä vahvasti soveltaen. Piirrustuksista poiketen aluksesta päätettiin tehdä pienempi, nykykäyttöön soveltuvampi ja kahden tykin sijasta vain keulatykillinen. Tykille ei myöskään tehty rataa jolla sen voisi siirtää aluksen keskelle siirtopurjehduksen ajaksi.

Suomenlinnan tykkisluuppi projektin tarkoituksena on

> ”työllistää ja kouluttaa nuoria, säilyttää ja elvyttää puulaivojen perinteisiä rakennustaitoja sekä kehittää ja elävöittää Suomenlinnan historiaa ja matkailua.” (Suomenlinna 2014.)

Oman projektini laiva perustuu enemmän af Chapmannin piirrustuksiin kuin Suomenlinnassa laivapuuseppä Mika Holmströmmin rakentamaan tykkisluuppiin. Käyttämäni laivamalli on pidempi ja kaksitykkinen jonka tykit pystyy siirtämään laivan keskelle siirtopurjehduksen ajaksi.

4 Serious games

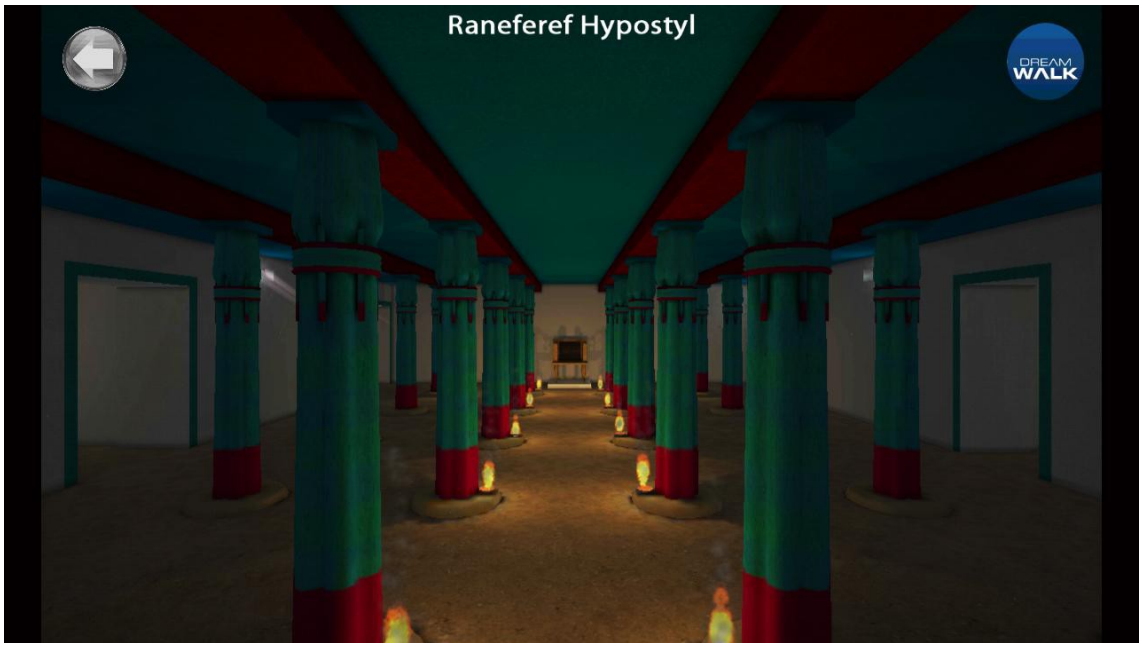
Tässä luvussa käsittelen hieman projektini genreä ja muita genreen kuuluvia applikaatioita. Käyn läpi genren yleisimpiä ominaisuuksia ja vertailen muita applikaatioita omaani. Haen applikaatioiden hyviä ja huonoja puolia ja kerron miten ne antoivat minulle ideoita omaan projektiini.

Serious games eli vakavat pelit on genre, joka ei pääasiassa tavoittele pelien viihteellisyyttä, vaan enemmän havainnollistamista ja opetusta (Adams 2009). Usein genrenä vakavat pelit ovat simulaatioita, joissa yritetään näyttää tai opettaa oikean maailman tapahtumia ja asioita ongelman ratkonnan tai havainnollistamisen kautta. Vakavia pelejä näkyy usein esimerkiksi lääketieteellisten tai puolustusalojen yritysten käytössä. Vakavia pelejä käytetään paljon opetuksen havainnollistamiseen ja markkinoinnin yhteydessä. Koska serious games -genren projektit eivät ole välttämättä pelejä, tulen kutsumaan niitä ja omaa projektiani applikaatioksi.

Koska omassa applikaatiossani ei ole erityisiä tavoitteita tai tehtäviä ja tarkoituksena on vain auttaa havainnollistamaan tykkisluuppia, se sijoittuu hyvin tähän serious games -genreen. Tämä olikin tietoinen päätös projektin alusta asti. Halusin tehdä tykkisluupista havainnollistavan interaktiivisen applikaation, joka kuuluisi kyseiseen kategoriaan. Tämän takia yritin etsiä muita genreen kuuluvia applikaatioita, jotta voisin tukimalla niitä keksiä genren yleisimpiä heikkouksia ja vahvuuksia. Näiden miettimisen jälkeen halusin parantaa tuloksilla omaa projektiani ja verrata yhteen omaa ja muiden tuotoksia.

4.1 Muita serious games -applikaatioita

Ensimmäinen löytämäni kyseisen genren applikaatio on tabletilta käytettävä Raneferef's Hall. Siinä rekonstruktioidaan osaa Egyptin kuningas Raneferefin hautakammiosta. Ensimmäinen hyvä asia mitä löysin projektista on useammat tavat katsoa 3D rekonstruktiota. Rekonstruktion sisällä pääsee liikkumaan näpäyttämällä maata kohdasta, johon haluaa kulkea, ja raahaamalla sormea näytöllä voi katsoa eri suuntiin. Sen lisäksi applikaatiossa on orbit-mahdollisuus joka vastaa enemmän oman projektini navigointia, eli mallia pääsee tarkastelemaan ulkopuolelta ja kamera pyörii sen ympärillä. Applikaatiossa on myös Dream View -mahdollisuus tabletilla, eli käyttäjä pääsee rekonstruktion keskelle ja katsomaan ympärilleen käyttäen tabletin gyroskooppia. Eli kun katselet tabletilla ympärillesi kamerakulma vaihtuu applikaatiossa ja näet mallin sisälle ikäänkuin ikkunasta katsottuna. Käyttäjälle on myös tarjolla videoita ja slide show -tyylinen esittely itse projektista ja projektin kohteen historiasta.



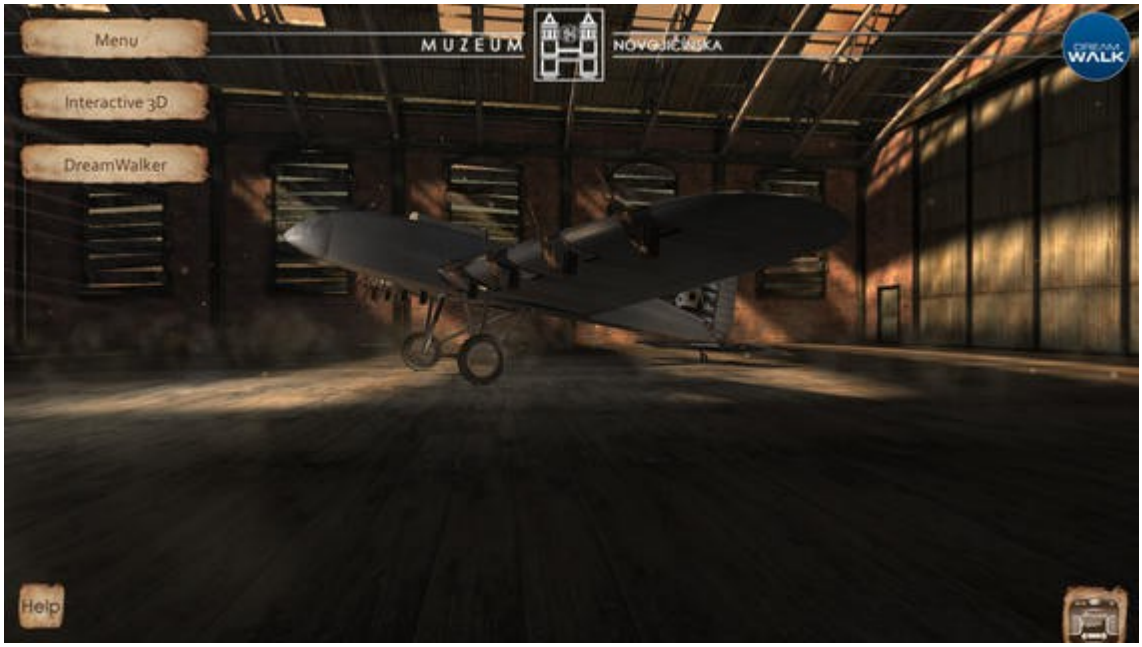
Kuva Raneferef's hall applikaatiosta (corinth 2013a).

Graafisesti applikaatio on melko heikko ja tutkittava tila on yksinkertainen sekä pieni. Geometria on hyvin yksinkertaista ja tekstuurit häiritsevän pieniresoluutioisia. Orbit-tavalla katsottuna täytyy tietysti nähdä mallin sisään joten seinien ulkopinnoista näkyy läpi, mutta tämä aiheuttaa myös pieniä graafisia kömmähdyksiä, kuten rakennusta kantavien tolppien puuttuvia pintoja niiden ylä- ja alapäissä.

Toinen löytämäni applikaatio on myös iOS-tabletilla toimiva MiracleAircraft. Applikaatio pyrkii visualisoimaan vuonna 1928 Vilém Zurovecin suunnittelemaa lentokonetta, jonka oli tarkoituksena pystyä koostaan huolimatta lähtemään lentoon lyhyiltäkin kiitoradoilta.

Applikaatiosta löytyy kolme eri tapaa tarkastella lentokonemallia. Käyttäjä voi tarkastella konetta lennossa kameran liikkuessa lentokoneen ympärillä. Lentokone on tällöin liikkeessä ilmojen halki propellit pyörien. Graafisesti lentokone on melko hyvin toteutettu ja jopa lentäjien kaulaliinat on animoitu lepattamaan tuulessa. Lentokone seuraa omaa animoitua reittiänsä koko ajan ympäristön yllä, joka on mukava kosketus koko hommaan. Ympäristö toisaalta on melko karu. Se koostuu geometrialtaan yksinkertaisesta maastosta jossa on jonkinlainen satelliitti tai ilmakuva tekstuurina oikeasta maastosta metsineen ja kylineen. Ideana ympäristö on hyvä, mutta sen yksinkertainen geometria ja kokonsa takia hyvin pieni resoluutioiseksi jäävä tekstuuri näyttävät aika karmeilta. Tästäkin applikaatiosta löytyy DreamView -ominaisuus jolla käyttäjä pystyy lennon aikana katselemaan ympärilleen koneen ohjaamosta.

Toinen tapa tarkastella konetta on maan päällä lentokonehallissa. Tässäkin ominaisuudessa kamera toimii orbit-mallisesti koneen ympärillä tai toisella DreamWalker-ominaisuudella jossa käyttäjä pääsee ensimmäisessä persoonassa kulkemaan lentokoneen ympärillä klikkailemalla maahan. Graafisesti itse halli on hyvin tehty ja lentokoneestakin on käytössä tarkempi malli. Tila on muutenkin valaistukseltaan mukavan tunnelmallinen.



Kuva MiracleAircraft-applikaatiosta (corinth 2013b).

Kolmas tapa tarkastella konetta on erillinen DreamView jossa on neljä eri valmiiksi renderöityä kuvaa lentokoneesta eri paikoista kuvattuna ja eri taustoissa. Tässäkin pystyy käyttäjä katselemaan ympäristöön tabletin gyroskooppia käyttäen. Kuvat ovat mielestäni vain liian pieniresoluutioisia, joten paikoittain ne ovat hyvinkin rosoisia.

Kokonaisuutena applikaatio on kyllä hienosti tehty. Tykkään itse lentokoneen mallista vaikka se onkin yksinkertainen, sillä se on tehty huolella ja se näyttää hyvältä. Plussana myös monta eri tapaa ja useampi ympäristö missä konetta voi tutkia. Tykkäsin myös paljon projektin navigointihelpistä. Help-menu koostuu vain videokameralla kuvatusta videosta, jossa näytetään kuinka navigointi toimii sormella ruudulla raahatessa ja tabletin gyroskoopin toiminta liikuttamalla tablettia ja näyttämällä mitä tapahtuu. Video on hyvin yksinkertainen, mutta näyttää tehokkaasti miten applikaatiossa navigoidaan.

Pöytäkoneelle oli hankalampaa löytää genren pelejä. Sain selville että Suomessa on yritys nimeltä Serious games Finland, mutta en löytänyt heidän tuotteistaan mitään mediaa saatika itse kokeiltavaa versiota. Unity 3D:llä on oma serious games -pakettinsa, jota he markkinoivat alan yrityksille, ja heidän sivustoiltaan löytyy esittelyjä projekteista ja tähän genreen liittyvät saa näkyviin non-games-napilla.

Tästä yhteenvetona huomasin että tablettiversio omasta työstäni olisi ollut hyvä lisä opinnäytetyöhöni. Tykästyin myös MiracleAircraft-applikaation navigointivideoon sen verran, että päätin toteuttaa omaan projektiini siitä hieman sovelletun version. Laivamallini olisi myös kaivannut hieman viilausta ja olisi ollut hieno saada sille ainakin yksi kunnon ympäristö missä sitä voisi tarkastella. ympäristöiksi olisi hyvin soveltunut esimerkiksi kuiva telakka tai jonkinlainen saaristoympäristö.

5 Reaaliaikamoottorin valinta

Ennen töiden aloittamista piti valita reaaliaikamoottori jolla opinnäytetyö toteutettaisiin. Ensimmäisenä selvitin Unity-moottorin soveltuvuutta projektiini, koska ajattelin että olisi mukavaa samalla opiskella itselleni täysin uutta moottoria. Ongelmana kyseisessä moottorissa oli sen ilmaisversion karsitut ominaisuudet. Ilmaisversiossa olivat esimerkiksi valaistusominaisuudet erittäin karsittuja. Reaaliaikaista valaistusta ei saanut eikä edes heittovarjoja. Heittovarjot olisi vielä voinut yhdistää tekstuureihin 3D-ohjelmassa, mutta ajattelin sen olevan liian iso urakka käyttääkseni Unity moottoria.

Lopuksi päädyin käyttämään Unreal 3 -moottoria. Tiesin moottorin käytöstä valmiiksi jo jotain joten sitä ei tarvinnut aivan alusta alkaen opiskella. Myös sen lisenssiehtojen takia moottorissa oli kaikki ominaisuudet käytettävissä ilman maksuja kunhan sillä toteutettu projekti ei ole maksullinen. Jos Unreal moottorilla haluaisi tehdä maksullisen pelin joutuisi moottorista maksamaan lisenssimaksut ja Epic-yhtiö ottaisi tuotoista tietyn osuuden. Unreal-moottorin noodipohjainen Kismet koodaus oli myös iso etu. Pystyin tekemään projektiini melkein kaiken toiminnallisuuden kirjoittamatta itse koodia joten sitä ei tarvinnut alkaa opiskelemaan.

6 Käytännön osuus

Seuraavaksi käyn läpi opinnäytetyön käytännön osuuden työprosesseja ja tekemistä. Joistakin työn osista kuten UDK-puolesta kerron hieman laajemmin ja menen syvemmälle prosessiin, ja jätän esimerkiksi laivamallin selittelyn hieman vähemmälle.

6.1 Optimointi

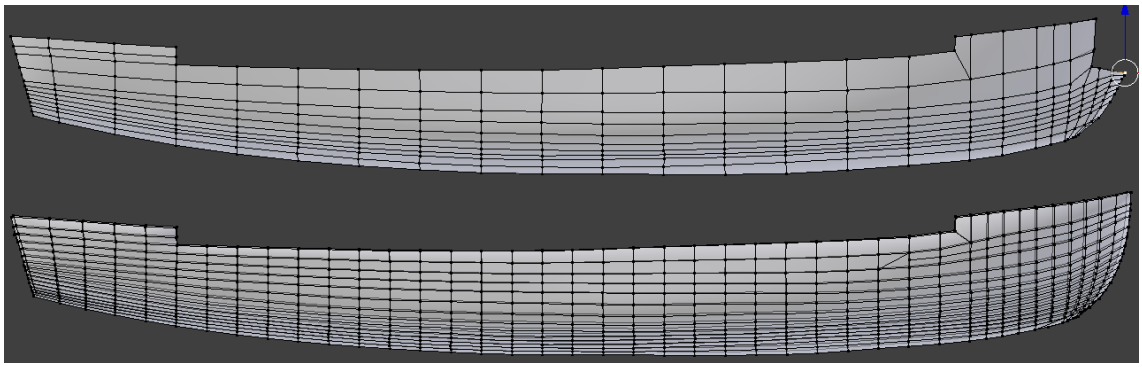
Jotta applikaatio toimisi sulavasti mahdollisimman monella koneella, piti assetteihin tehdä hieman optimointeja. Mallin topologiaa piti yksinkertaistaa ja miettiä tekstuurien kokoa. Applikaationi scene on kuitenkin niin pieni ja yksinkertainen ettei minun tarvinnut tehdä suuria uhrauksia esimerkiksi tekstuurikokojen kanssa.

6.2 Tekstuurit

Suurin osa tekstuureistani on 1024 kertaa 1024 pikselin kokoisia, mutta joitakin 2048 kertaa 2048 resoluutioisiakin tekstuureita löytyy scenestä. Jos tekstuurikoosta olisi pitänyt alkaa tinkimään olisin aloittanut sen spekulaaritekstuureista. Tällä hetkellä kaikki materiaalini spekulaaritekstuurit ovat saman kokoisia kuin diffuusitekstuuritkin ja niiden pienentämisellä ei olisi ollut suurta vaikutusta lopputulokseen.

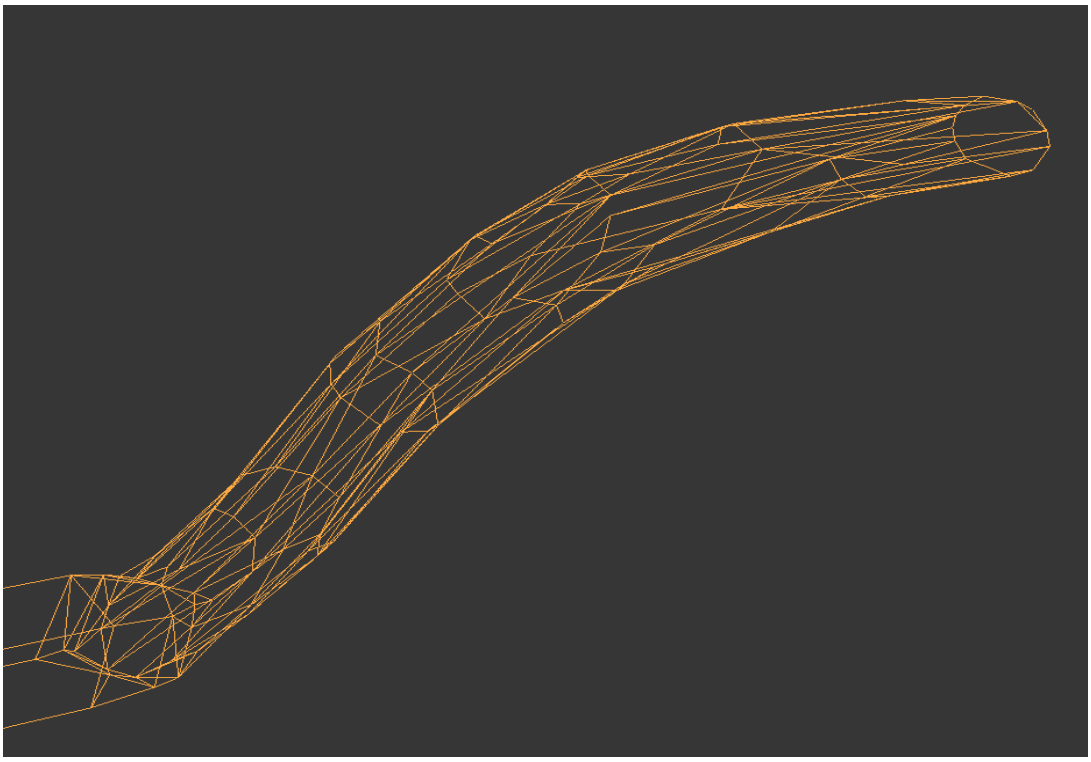
6.3 Mallin topologia

Käytin työssäni aikaisemmin tehtyä laivamallia, jota oli käytetty tykkisluupin promotionaalimateriaalien renderöintiin. Malli oli kuitenkin liian korkeapolygoninen reaaliaikamoottorissa pyöritettäväksi joten sitä piti ensin yksinkertaistaa. Ennen topologian yksinkertaistamista malli koostui noin 246 tuhannesta facesta ja sain määrän vähennettyä noin 15 tuhanteen faceen uhraamatta kauheasti laivan yksityiskohtia.



Kuvassa esimerkki optimoidusta geometriasta. Ylhäällä optimoitu ja alhaalla optimoimaton versio samasta rungosta.

Ensin poistin mallista hyvin pieniä yksityiskohtia, jotka pystyin myöhemmin toteuttamaan tekstuureina. Pienten detaljien karsimisen jälkeen lähdin pala palalta yksinkertaistamaan mallin geometriaa. Poistin turhia edgelooppeja, jotka joko eivät vaikuttaneet mallin muotoon, tai vaikuttivat tarpeeksi vähän, että niiden poistaminen ei muuttanut mallin siluettia liian dramaattisesti. Muutaman palasen geometria oli niin epäloogista että mallinsin ne kokonaan uudestaan pitäen vanhaa palaa referenssinä.



Yksi epäloogisista objekteista

Näiden muutamien palasten geometria vaikutti korruptoituneelta, mikä voi aiheutua esimerkiksi joistain export tai import -toiminnoista.

6.4 Mallin valmistelu ASE exportointia varten

Seuraavaksi käyn läpi 3D-mallien exportointia Unreal-moottoria varten, mitä siitä täytyy tietää ennen exportointia ja miten siihen kannattaa valmistautua. Unreal Editor pystyy importoimaan 3D-malleja .ASE ja .FBX -tiedostoina. ASE-tiedostoina saa helposti importoitua itse 3D-mallit, mutta jos haluaa sisällyttää 3D-ohjelmassa tehdyt animaatiot importoitavaan tiedostoon täytyy käyttää .FBX-tiedostotyyppiä. Tähän tiedostotyyppiin sisällytetyt animaatiot löytää ja voi aktivoida importoinnin jälkeen Unrealin Matinee -editorista omalta raidaltaan.

Ennen 3D-mallien exportointia Unreal Editoriin kannattaa harkita eri objektien yhdistelyä 3D-ohjelmassa jotta saataisiin mahdollisimman vähän piirtokutsuja Unreal-moottorissa, muistaen kuitenkin pitää erillään objektit joilla on tarkoitus olla eriäviä funktioita kuten piilottaminen ja animoituminen. Useamman identtisen objektin kohdalla täytyy päättää lisääkö ne kaikki samaan objektiin vai importoiko Unreal Editoriin vain yhden kappaleen niitä ja instansoi niistä tarvittavan määrän itse pelimoottorissa. Objektien instansoiminen säästäisi resursseja. Airojen tapauksessa järkevämmältä tuntuu vain yhden kappaleen tuominen pelimoottoriin ja sen instansointi. Instansoimalla airoja niiden mahdolliset animaatiot voisi suoraan kopioida airolta toiselle. Toisaalta jos airot haluaa piilottaa täytyy hide-komento tehdä Kismetissä jokaiselle airolle erikseen.

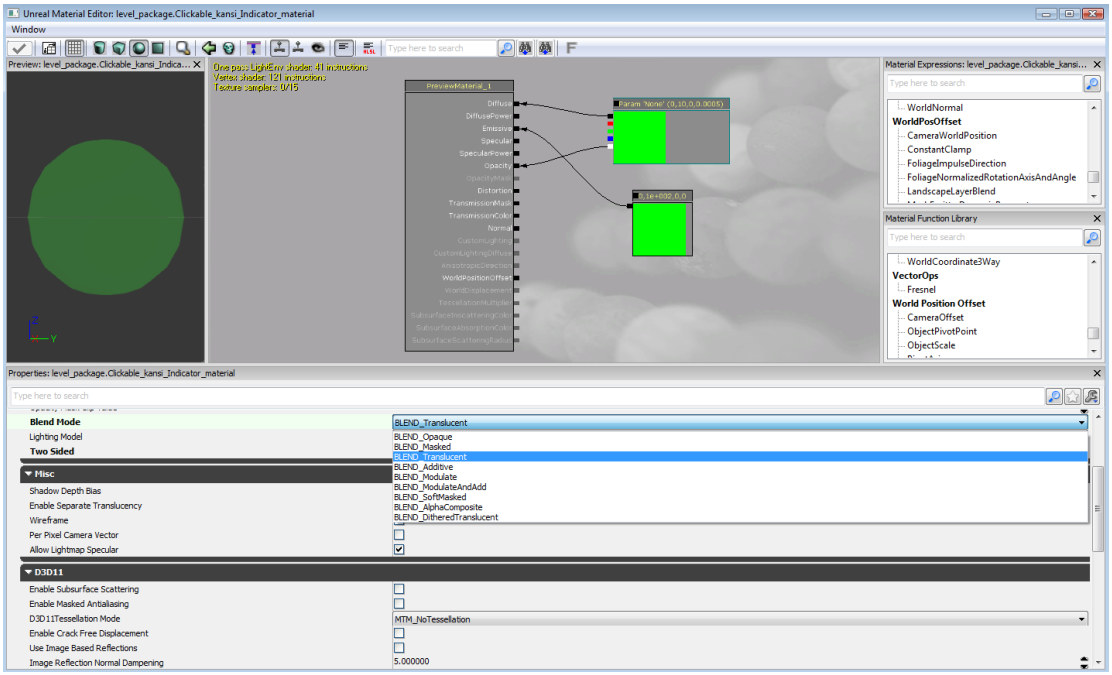
On hyvä pitää mielessä että ainakin blenderin .ASE-exportoinnissa pivot pisteet liikkuvat aina origoon, joten Unreal Editorissa kannattaa myöhemmin säätää pivot pisteiden paikkaa. Pivot pisteiden paikkaa voi säätää oikea klikkaamalla haluttua objektia ja painamalla pivot nappia. Nyt pivotin paikkaa voi säätää normaaleilla työkaluilla. Kun pivot piste on oikeassa paikassa sen voi tallentaa painamalla save prepivot to pivot nappia. (KdgDev 2011.) Jos objektien tarkoitus on olla osa isompaa kokonaisuutta voi olla hyvä pitää pivot pisteet samassa paikassa, jotta palasten paikat löytyvät helposti. Mutta jos objektia on tarkoitus animoida, tai sitä tullaan instasoimaan, voi olla hyvä viedä objekti origoon, jolloin exportoinnin jälkeen sen pivot piste on sen geometriassa. Blenderin .ASE-exportointia varten mallia täytyy hieman valmistella. kaikki meshit täytyy olla UV-unwrapattu ja niillä täytyy olla jonkinlainern materiaali (Mcampagnini, Richisbored 2012).

Myöhemmin tehdessäni materiaaleja laivan palasille Unreal Editorissa huomasin muutaman .ASE-exportoinnin yhteydessä tapahtuneen virheen. Esimerkiksi joidenkin palasten muutaman facen normaalit olivat kääntyneet väärinpäin, ja vaikka materiaalit olivat kaksipuoleisia, ne tekivät häiritsevän näköisiä artifakteja mallin pinnoille. Ongelman ratkaisin korjaamalla normaalit blenderissä ja ennen .ASE-exportointia tarkistin, että recalculate normals -vaihtoehto oli kytketty pois päältä. Korjatut mallit piti vain importoida Unreal Editoriin uudestaan, ja nyt normaalit toimivat hyvin ja artifaktit olivat kadonneet.

6.5 Teksturointi ja materiaalit

Laivan retopologian tekemisen yhteydessä tein sen palasille UV-kartat blenderissä. Tekstuurit maalasin photoshopissa, ja tein materiaalit valmiiksi myös blenderissä, jotta saisin laivasta 3D-ohjelmalla renderöityjä kuvia ja videoita. Laivan materiaalit koostuvat tekstuurikartoista ja spekulaarikartoista. Suurin osa laivan tekstuureista on 1024 kertaa 1024 pikselin kokoisia, mutta muutamassa objektissa on myös 2048 kertaa 2048 pikselin kokoiset tekstuurit. Kun laivan materiaalit olivat valmiita blenderissä, piti ne viedä vielä Unreal Editoriin lopputuloksessa käytettäväksi. Importoin level pakettiin jokaisesta laivan palasesta tekstuurikartan ja spekulaarikartan. Nämä kartat yhdistin Unrealin materiaalieditorissa materiaaleiksi jotka annoin niille kuuluville palasille itse scenessä.

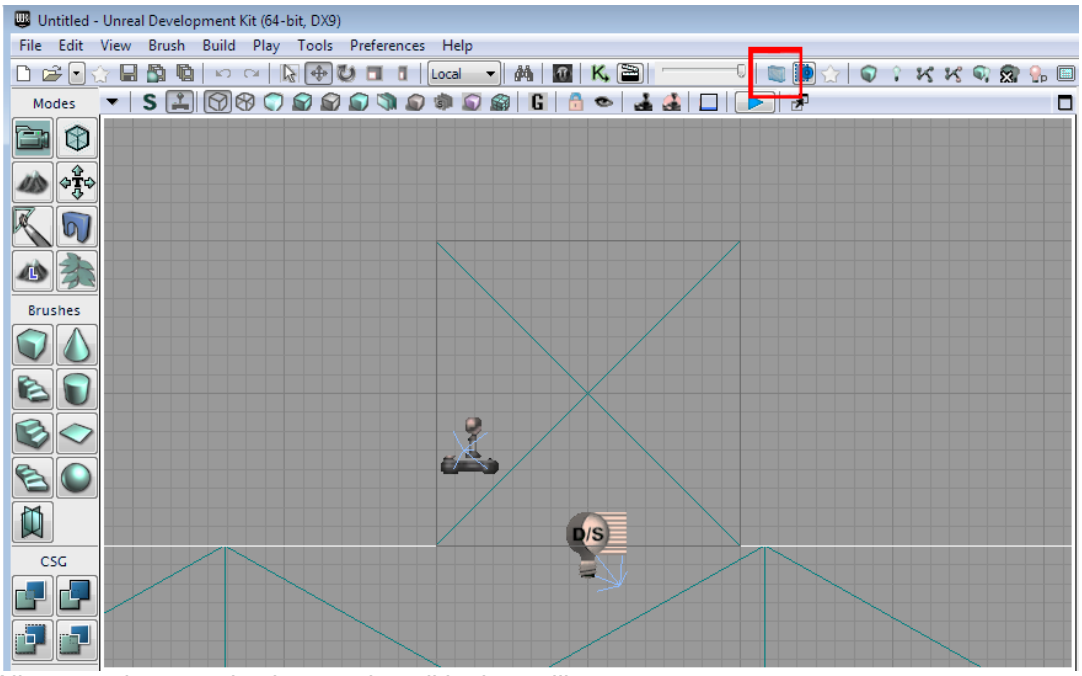
Unreal Editorin materiaalieditori toimii muiden Unreal Editorien kaltaisesti noodeilla. Kismetistä poikkeavasti material editorissa informaatio kulkee oikealta vasemmalle ja yhdistyy yhteen previewmaterial-noodiin. Toin tekstuurikartan ja spekulaarikartan materia editoriin texture sample -noodeina ja yhdistin ne previewmaterial-noodin diffuse ja specular -kohtiin. Joissain materiaaleissa kuten laivan purjeissa piti myös määritellä ne kaksipuoleisiksi materiaalien asetuksista. Laivan interaktiivisuusindikaattori palasten materiaalit on tehty läpinäkyviksi määrittelemällä niiden blend mode BLEND_translucentiksi. Blend modet määrittävät miten materiaalin värit yhdistyvät taustan väreihin.



Material editorin valikot

Tämän blend moden valitsemalla previewmaterial-nodeen aktivoituu uusi opacity-kanava. Tein läpinäkyviin materiaaleihin uuden VectorParameter-noodin jossa pystyin määrittelemään materiaalin värin ja sen alpha-kanavasta läpinäkyvyyden ja kytkin ne previewmateriaali-noodin diffuse ja opacity -kanavoihin.

Läpinäkyvien materiaalien kanssa työskennellessä kannattaa harkita käyttöliittymän ikkunasta Allow translucent selection -napin päälle kytkemistä.



Allow translucent selection -napin paikka käyttöliittymässä

Unreal Editor normaali asetuksillaan ei anna käyttäjän valita 3D viewportista actoreita joilla on läpinäkyvä materiaali. Valinta onnistuu kyllä wireframe viewporteista, mutta varsinkin monimutkaisemmissa sceneissä se saattaa olla hankalaa.

6.6 Unreal Kismet

Unreal Kismet on moottorin oma visuaalinen noodipohjainen scriptaus-systeemi. Se on tehokas työkalu, jolla käyttäjä pystyy vaikuttamaan miten hänen scenensä ja jopa pelinsä toimivat kirjoittamatta lainkaan koodia. Kismet koostuu yksinkertaisista noodeista ja niitä yhdistävistä linjoista, joilla voi rakentaa monimutkaisiakin tapahtumaketjuja tai interaktioita pelissä. Intuitiivisuutensa takia kismet on yksi syy jonka takia valitsin Unreal Enginen projektini moottoriksi. Siitä on myös tullut iso osa opinnäytetyötäni koska suurin osa projektin toiminnoista on toteutettu Kismetillä.

Kismetissä informaatio välittyy noodien välillä vasemmalta oikealle eli noodeissa on usein input-kanavat vasemmalla puolella ja output-kanavat oikealla. Koska suurempien levelien noodirakenteet saattavat paisua suuriksi ja sekaviksi, on Kismettiin lisätty myös mahdollisuus järjestellä ja nimetä noodeja ja rakenteita. Käyttäjä pystyy nimeämään tai kommentoimaan yksittäisiä noodeja ja kokonaisia noodi rakenteita. Nimeäminen onkin erittäin suositeltavaa, koska jo minun pienikokoisessa levelissä noodi rakenteet alkoivat olla melko suuria ja monimutkaisia. Selkeä rakenne ja nimeäminen on myös erittäin tärkeää jos useampi ihminen työstää saman levelin Kismet noodeja. Projektiini olen toteuttanut Kismetissä navigoinnin laivan ympärillä, interaktiivisuuden laivan osien kanssa, kamera ajot ja niiden käynnistämisen ja Flash käyttöliittymän yhdistämisen sceneen.

6.7 Levelpaketit ja ulkoiset paketit

Tässä luvussa kerron hieman Unreal Editorin pakettisysteemistä ja assettien hallinoinnista. Varsinkin suuremmissa projekteissa on erittäin tärkeää että assetit säilötään oikein omiin paketteihinsa, joten jokaisen Unreal-moottorin käyttäjän on hyvä tietää niiden hallinnasta.

Kaikki Unreal Editorin ulkopuolella tehdyt assetit, joita käytetään editorissa, säilytetään paketeissa. Kun käyttäjä importoi assetin editoriin, samalla kysytään että mihin pakettiin se halutaan säilöä vai tehdäänkö sille uusi ulkoinen paketti. Jokaisella levelillä on oma levelpakettinsa ja per level niitä voi olla vain yksi. Leveli voi sen lisäksi myös referensoida useampiakin ulkoisia paketteja.

Levelpaketti on tallennettu suoraan .udk-tiedostoon, joten sen referoimisessa ei tule ongelmia, mutta saattaa tehdä tiedostosta suurikokoisen. Ongelmana on myös, että jos samaa assettia käytetään useammassa levelissä, joudutaan se tallentamaan jokaiseen sitä käyttävän .udk-tiedoston sisäiseen pakettiin. Tämä on tiedostokoon tuhlausta.
Ulkoiset paketit tallenetaan erillisinä .upk-tiedostoina käyttäjän valitsemaan paikkaan. Tämä taas voi aiheuttaa ongelmia assettien referoimisessa, varsinkin jos leveltiedosto ei ole tallennettu UDK:n ehdottamaan paikkaan. (NATO_chrisjm 2012.) Ulkoisen paketin mukana käyttäjälle tulee vastuu niiden tallentamisesta ja eteenpäin antamisesta, jos työryhmässä on useampi samaa leveliä muokkaava ihminen.

Koska omassa työssäni käytän vain yhtä level tiedostoa ja assettien määrä ei ole suuri, päädyin aluksi level paketin käyttöön. Myöhemmin Flash-käyttöliittymää tehdessäni huomasin tehneeni sille vahingossa oman paketin, joten lopuksi päädyin käyttämään level paketin lisäksi ulkoista pakettia.

6.8 Navigaatio

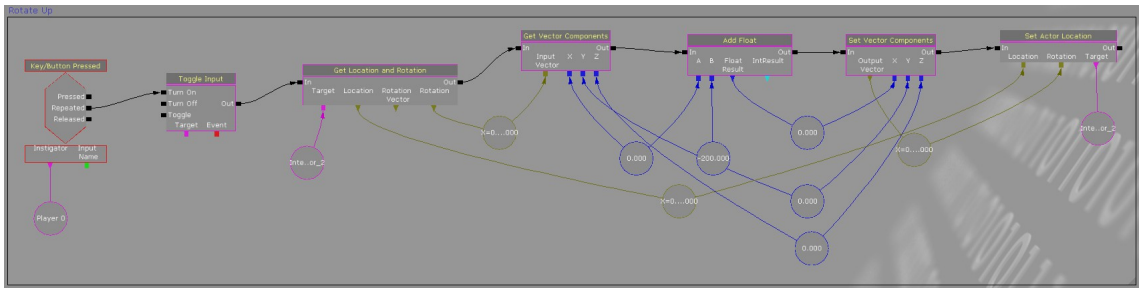
Saatuani laivamallin kokonaisuudessaan materiaalien kanssa sceneen oli aika ryhtyä implementoimaan projektin tärkeimpiä ominaisuuksia, joista ensimmäisenä navigointi scenessä. Halusin tehdä navigoinnista mahdollisimman intuitiivistä joten päädyin usein käytettyyn hiiren click and drag -navigointiin. Tässä luvussa kerron tarkemmin hiiren click and drag toiminnon toteutuksesta ja hiiren rullan avulla toimivasta zoomista.

Navigaation pääominaisuudet halusin toimivan pääosin hiiren, ja vaihtoehtoisesti näppäimistön, avulla. Kameran rotaatiot halusin toimivan hiirellä click and drag tyylisesti ja myös näppäimistön nuolinäppäimillä. Kameran zoomaus taas toimii helposti hiiren rullan avulla. Ennen kuin navigaatiota pystyi alkaa toteuttamaan, piti päättää rotatoikuuko laiva kameran edessä vai kamera laivan ympärillä.

Laivan rotatoitumisella kameran edessä olisi olisi ollut omat ongelmansa, kuten laivan eri osien animoiminen, joten päädyin kameran rotatoimiseen laivan ympärillä.

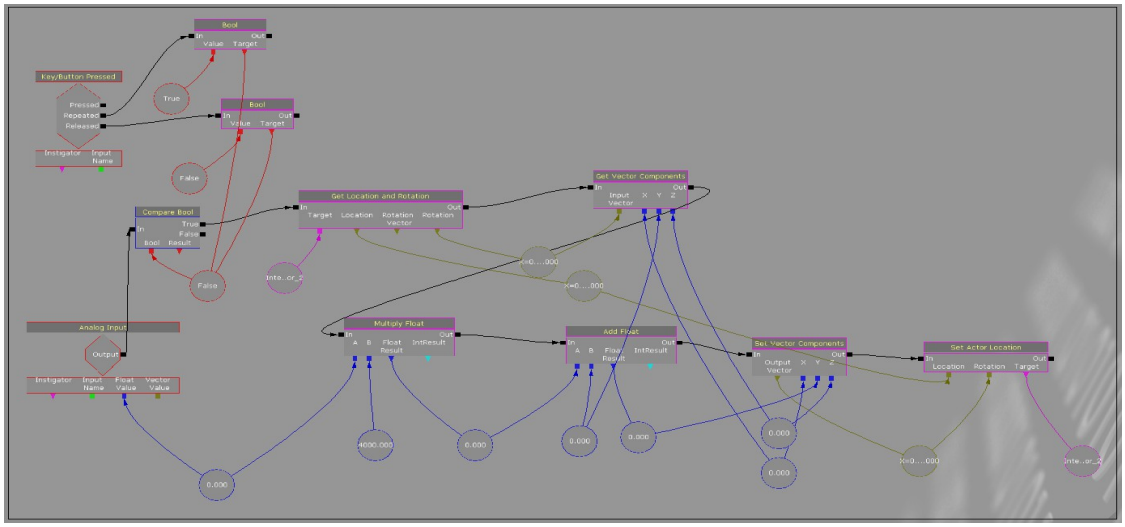
6.8.1 Kameran pyöriminen laivan ympärillä

Kamera on kiinnitetty levelissä Kismetin attach to actor -noodilla kuutiomeshiin ja se on myös annettu kameran kohteeksi, joten liikkuessaan kamera osoittaa aina kuutiota kohti. Kyseistä kuutiomeshiä sitten ohjataan syöttämällä Kismetissä sen rotaatioarvoihin uusia lukemia ja näin saadaan kamera liikkumaan kuution mukana. Kismetissä haetaan kameran kuution rotaatioarvot Get Location and Rotation -noodilla, ja sen tuottama vektorilista hajotetaan sen eri vektorikomponenteiksi Get Vector Components -noodilla. Kun vektorit on saatu omiksi float-arvoiksi, voi haluttuun rotaatiovektoriin lisätä Add Float -noodilla lukemia, ja syöttää ne taas kuution rotation arvoihin Set Vector Components- ja Set Actor Location -noodeilla. Nuolinäppäimillä rotatoimiseksi lisättiin Add Float -noodilla rotaatiosuunnasta riippuen 200 tai -200 arvo kuution rotatioon. Koko prosessi laukaistaan Key/Button pressed -noodilla joka aktivoituu halutun nuolinäppäimen painamisella. (ThatDutchGuyUDK 2013.)



Nuolinäppäimillä toimiva rotaatio

Hiiren klikkaus ja raahaus -navigaatiossa taas tarkistetaan jatkuvasti hiiren liikkuessa Compare Boolean -noodilla, onko hiiren vasen näppäin painettuna. Jos näppäin on painettuna, otetaan Analog Input -noodista hiiren liikkeen nopeus ja suunta float-arvona, joka lisätään kameran kuutiomeshin rotatoitumisarvoihin.

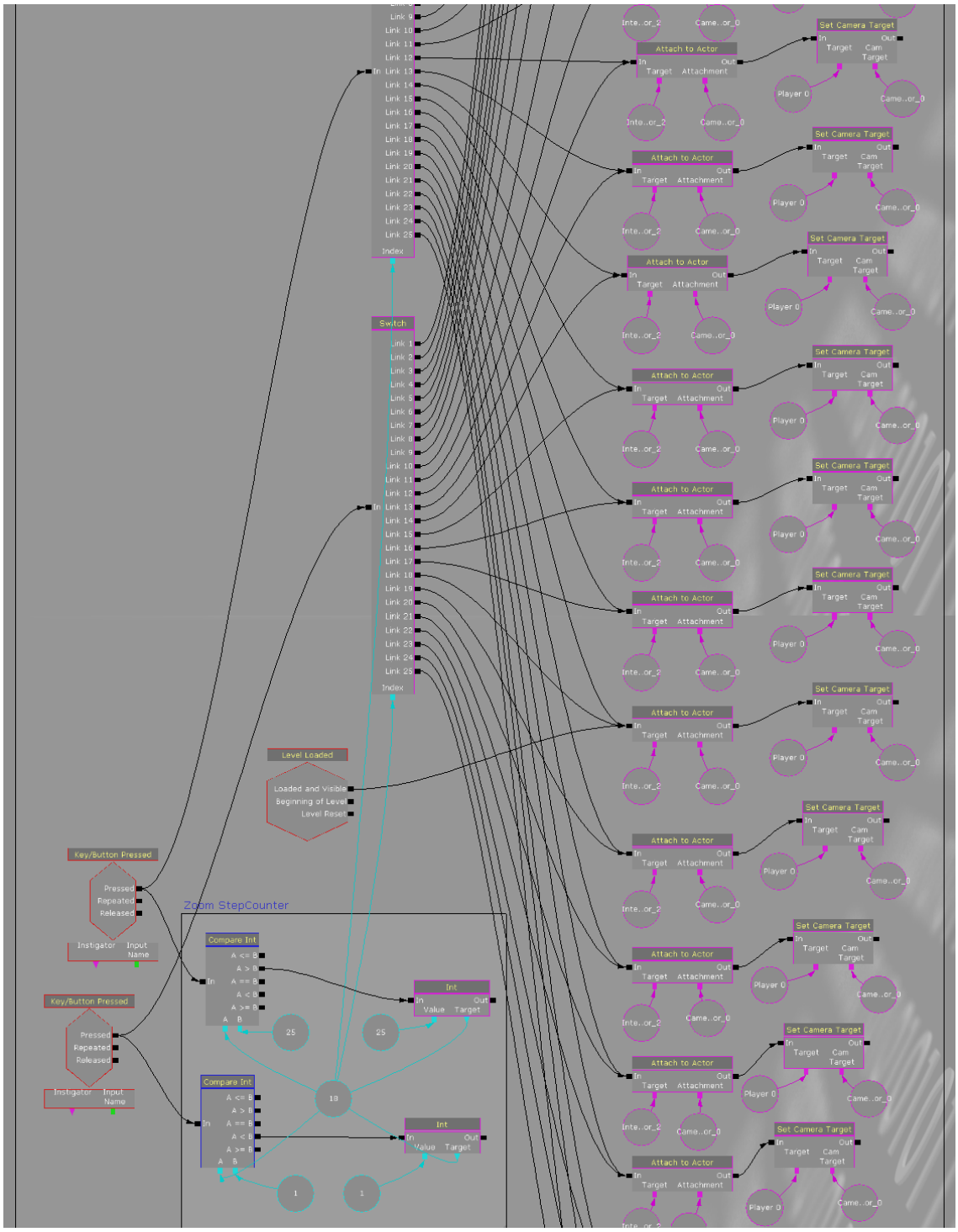


Hiirellä toimiva rotaatio

Hiirellä toimivassa rotaatiossa on pieni bugi, jota en saanut yrityksistäni huolimatta koskaan korjattua. Kameran pyöriessä laivan ympäri, joskus kamera saattaa yhtäkkiä hypätä aivan uuteen asentoon, vaikka käyttäjä olisikin tehnyt vain pienen hiiren liikkeen. Ongelma voi mahdollisesti johtua ruudunpäivityksessä tapahtuneesta töksähdyksestä, jonka aikana noodisysteemi ehtii lähettää suuremman hiiren liikkeen arvot kamerakuution rotaatioihin kuin normaalisti. Tarkasta syystä en kuitenkaan koskaan saanut varmuutta, joten ongelma jäi korjaamatta. Onneksi kyseinen tapahtuu hyvin harvoin ja se ei tapahtuessaan riko applikaatiota millään tavalla.

6.8.2 Zoom

Ulos ja sisään zoomaus on toteutettu monella Attach to Actor -noodilla, joilla kiinnitetään kamera sitä ohjaavaan kuutiomeshiin. Jokaisessa kyseisessä noodissa on oma etäisyytensä kameran ja kuution välillä. Näitä noodeja hallitaan kahdella Switch-noodilla joista toinen aktivoituu hiiren rullasta eteen, ja toinen taakse, rullattaessa. Kun Switch-noodi aktivoidaan, se vuorostaan aktivoi sille asetetun linkkilistan seuraavan tai aikaisemman linkin, joka aktivoi Attach to Actor -noodin, jossa on switchistä riippuen pienempi tai isompi etäisyys kameran ja sitä ohjaavan kuutiomeshin välillä.



Zoom ja integer arvon laskuri

Jotta Switch-noodit pysyisivät synkronoituna samassa linkissä, niiden indexiin on laitettu yhteinen Integer-arvo.

Ulkoisen indexin takia switch-noodit voivat myös liikkua pois omalta linkkilistaltaan, jonka takia sille piti tehdä laskuri, joka tarkastaa jatkuvasti Integer-arvon ja pitää sen linkkien lukumäärän rajoissa.

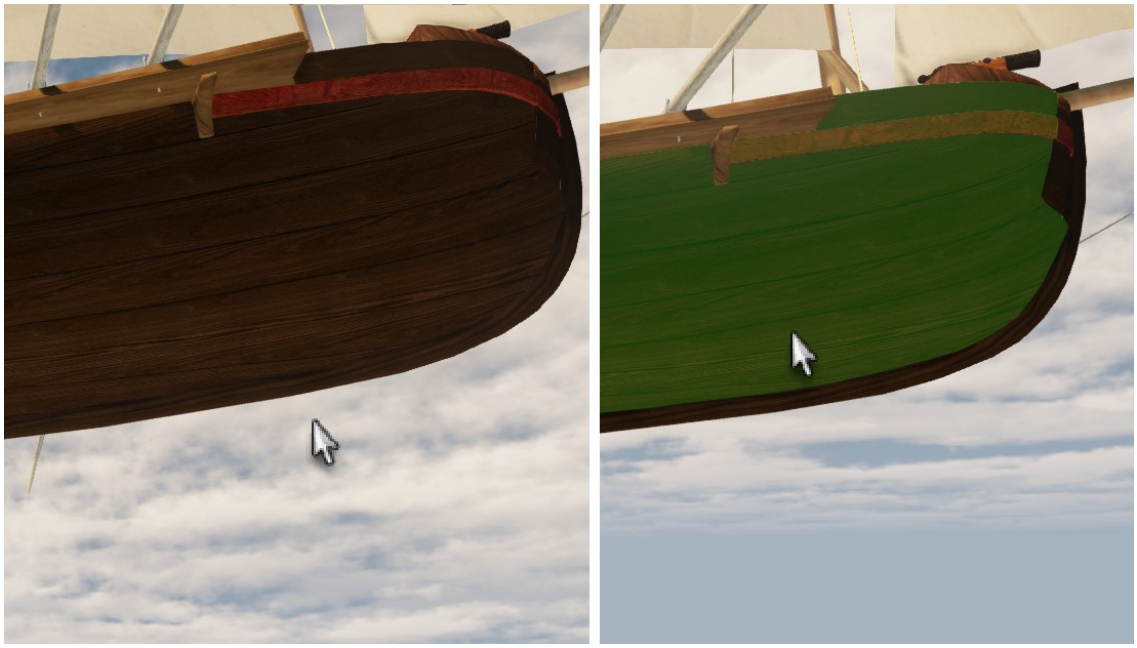
6.9 Interaktio laivan kanssa

Tarkoituksena oli tuoda laivan interaktiiviset osat esille kun hiiren kursori liikkuu niiden päälle. Kun interaktiivinen osa on korostettu voi sen piilottaa, näyttää uudelleen, tai aloittaa animaation klikkaamalla sitä.

Toimivan hiiri-interaktion aikaansaamiseksi Unreal moottorissa tarvitaan UnrealScript-kieltä. Sain valmiit scriptit Unreal Developer Networkista ja jäljelle jäi niiden sisällyttäminen moottoriin ja niiden implementointi työhön. Kun Unreal Engine lukee kyseistä scriptiä, löytyy editorin world properties valikosta uusi MouseInterFaceGameInfo pelimuoto, jota käyttämällä saadaan applikaation alkaessa ruutuun hiiren kursori. Pelimuodon lisäksi scripteissä tulee uusi MouseInterFaceKActor actor -luokka, joka mahdollistaa hiiri-interaktion kyseisen actor-luokan meshien kanssa. Actor-luokalle löytyy Kismetistä uusi noodi, jossa on inputit hiiren nappuloista, jotka toimivat pelissä kursorin ollessa actor luokan käyttämän meshin päällä. (Epic Games, inc. 2011.) Eli tällä noodilla voi esimerkiksi klikkauksella piilottaa meshin tai animoida sen materiaalia.

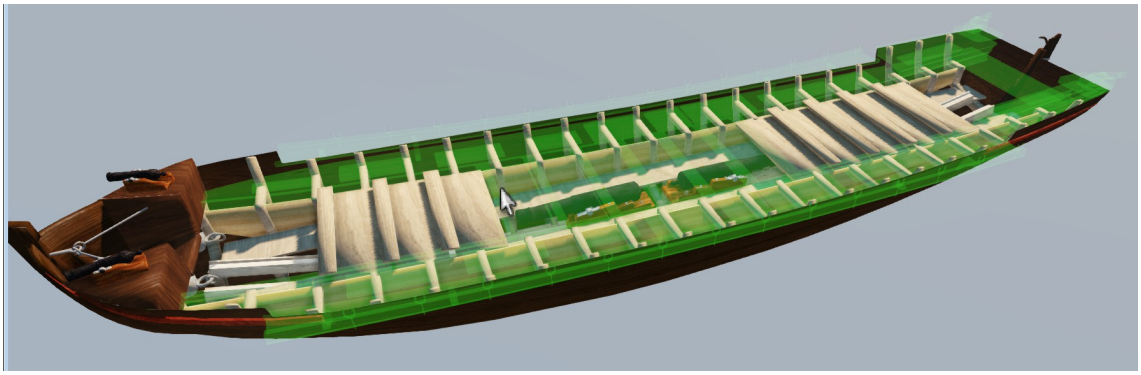
6.9.1 Laivan palasten korostaminen

Jotta laivan interaktiiviset osat erottuisivan muista, korostin niitä samanlaisilla osilla, joilla on vihreä läpinäkyvä materiaali. Nämä korostusobjektit tein muokkaamalla laivan osia hiukan isommiksi blenderissä, jotta niiden pinta näkyisi juuri ja juuri laivan vastaavan palasen pinnan päällä. Applikaation käynnistyessä kaikki nämä interaktioindikaattoriobjektit ovat piilotettuna, mutta ne muuttuvan näkyviksi kursorin ollessa sellaisen päällä.



Esimerkki Interaktiivisesta laivanpalasta

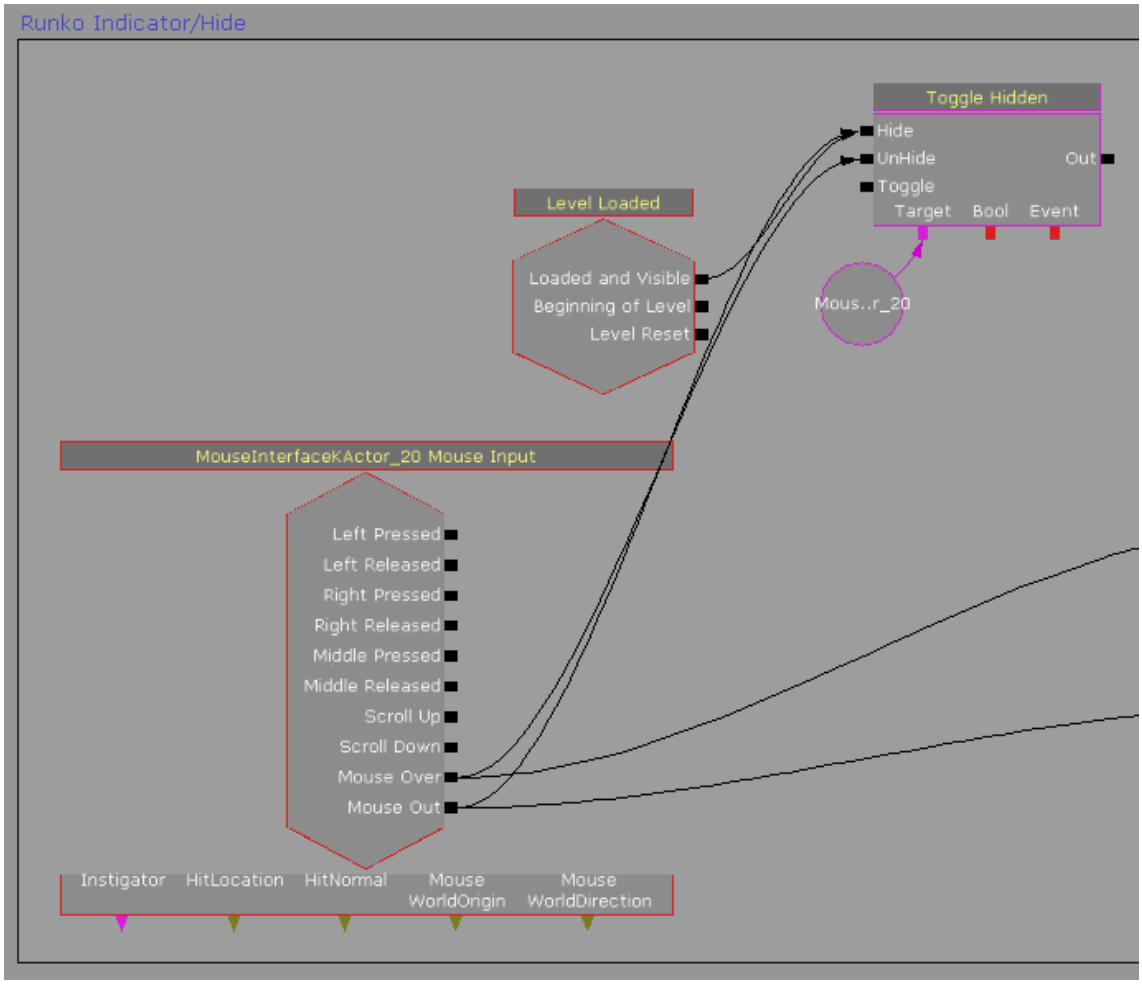
Päädyin tekemään interaktiivisten osien indikaattoreista omat geometriansa, enkä animoinut itse laivan palasten materiaalia, vaikka se olisi säästänyt polygoneissa ja draw calleissa. Halusin indikaattorit omaksi geometriakseen, koska silloin ne näkyisivät myös vastaavan laivan palasen ollessa piilotettuna, ja näin indikaattori luo haamumaisen kuvan kyseisestä palasesta samalla näyttäen käyttäjälle mahdollisuuden saada se esiin hiiren klikkauksella. Scenen yksinkertaisuudesta johtuen indikaattorien lisätyt geometriat, materiaalit ja draw callit eivät aiheuta suuria rasituksia tai vaadi isompia konetehoja.



Piilotetun palasen korostus

Jotta objekti voisi olla interaktiossa hiiren kanssa, sen täytyy olla MouseInterfaceKActor luokkaa. Tämä uusi objektiluokka on tullut MouseInterfaceGameInfo Unreal Scriptin kanssa.

Objektien näyttäminen ja piilottaminen hiiren kursorilla on tehty Mouse Input -noodilla. Noodi on liitetty mouse over kohdasta Toggle Hidden -noodin unhide kohtaan, ja mouse out kohdasta hide kohtaan. Toggle Hidden -noodin targetiksi pitää laittaa haluttu objekti object variablena, ja jotta indikaattori ei olisi näkyvillä applikaation alkaessa, Toggle Hidden -noodiin tulee myös Level Loaded -noodin loaded and visible kohdasta kytkös hide kohtaan.

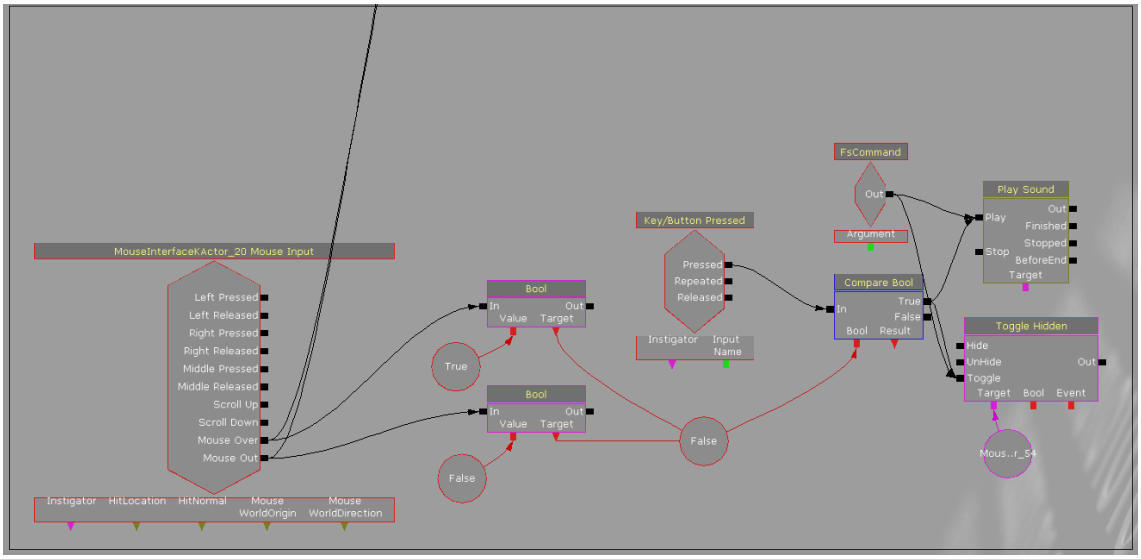


Mouse Input nodella piilotettava objekti

Kaikki laivan interaktiot on myös aktivoitavissa käyttöliittymän nappuloista. Nappuloista voi myös aloittaa laivan läpi kulkevan kamera-ajon, nähdä ohjeet navigoinnille ja lukea infoa työstä. Käyttöliittymä on tehty Flash ohjelmalla ja sen tekemisestä kerron tarkemmin toisessa osiossa.

6.9.2 Laivan palasten piilottaminen ja näyttäminen

Laivan palasten piilottaminen ja näyttäminen onnistuu hiiren klikkauksella kun kursori on kyseisen palasen päällä. Kun Kismet huomaa käyttäjän painavan hiiren vasenta nappulaa se tarkistaa Compare Boolean -nodeilla onko kursori jonkun interaktoitavan laivan osan päällä. Jos kursori on jonkin tälläisen osan päällä, Kismetissä menee singaali eteenpäin kyseisen osan Toggle Hidden -noodin toggle kohtaan. näin aktivoituna Toggle Hidden -noodi tarkastaa osan nykyisen tilan, eli onko se piilotettuna vai näkyvänä pelaajalle. MouseInterfaceKActor luokan objektit huomaavat hiiren kursorin ja ovat interaktiivisia vaikka ne olisivatkin piilotettuina. Riippuen sen tilasta noodi vaihtaa sen tilaa toisenlaiseksi, eli jos osa on piilotettu, se vaihdetaan näkyväksi.



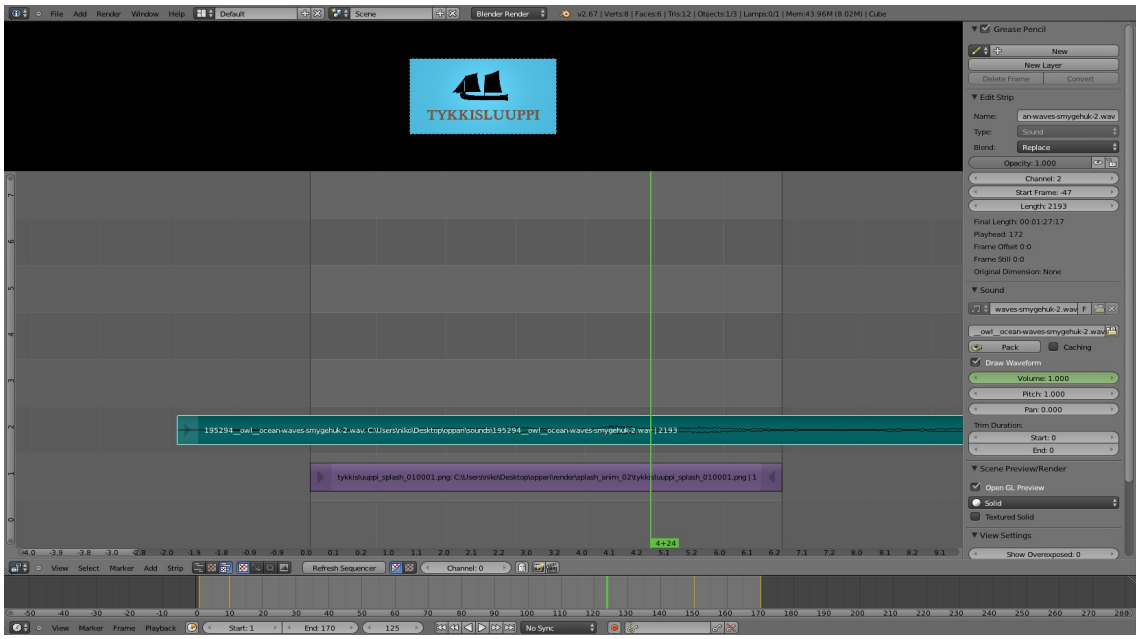
Osien piilotus ja näyttäminen Kismetissä

Toggle Hidden -noodin lisäksi Compare Boolean lähettää viestin myös Play Sound -noodiin. Näin siis jokaisen interaktiiviseen osaan osuvan napinpainalluksen yhteydessä kuuluu ääni.

6.10 Äänet

Äänet ovat tärkeä osa applikaatiota sen lopputuloksen uskottavuuden kannalta. Esimerkiksi hiiren klikkauksesta aktivoituvat äänet antavat hyvää painoarvoa jokaiselle merkitykselliselle klikkaukselle ja taustaäänet taas luovat helposti syvyyttä tyhjemmällekkin scenelle.

Ääniä projektissa tuli splash-animaatioon, ambient-taustoiksi ja klikkausääniksi itse applikaatioon. Splash-animaation äänet yhdistin renderöityyn videoon blenderin omassa videosekvenssieditorissa. Äänien yhdistämisen lisäksi valitsin ääniraidasta sopivan kohdan splash-animaation aikana soitettavaksi. Tein ääniraitaan fade in ja fade out efektit, jotta äänet eivät ailkaisi ja loppuisi yhtäkkisesti.
Äänien yhdistämisen ja säätämisen jälkeen renderöin lopullisen splash-animaation ulos videosekvenssieditorista.



Blenderin videosekvenssieditori

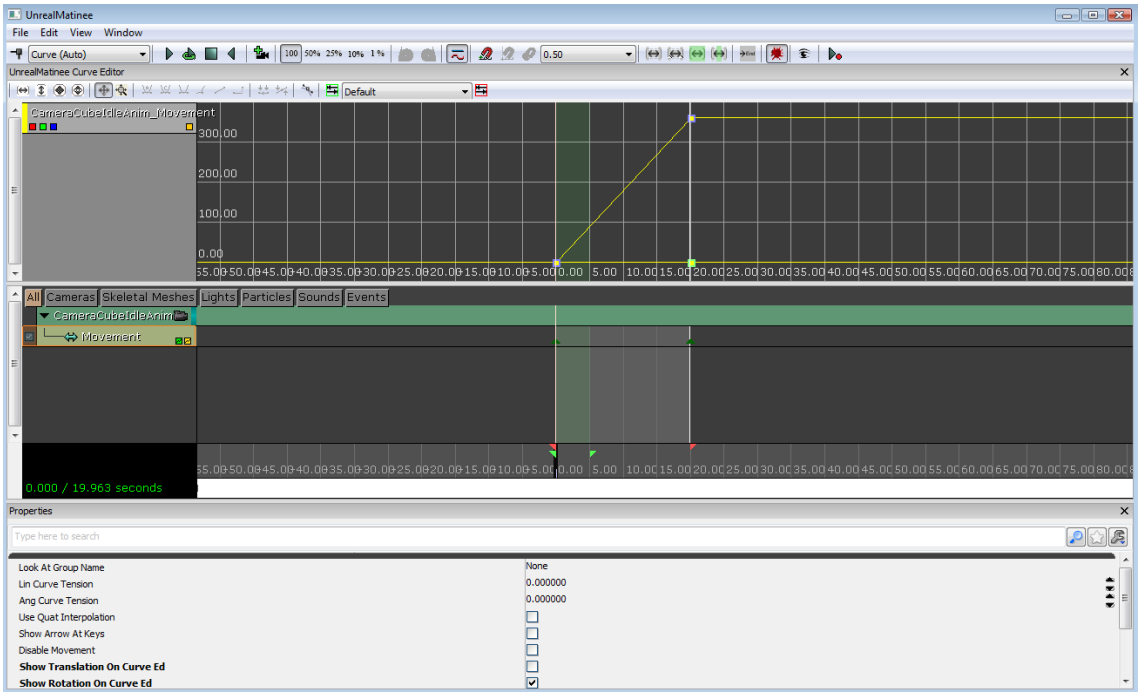
Muissa tapauksissa äänet piti implementoida Unreal Enginessä. Äänien tuonti Unreal Editoriin toimii normaalisti import toimintona ja toin ne kaikki sinne .wav tiedostoina. Äänien importoinnin yhteydessä tein jokaiselle äänelle oman SoundCuen täyttämällä "Auto create sound cue" kohdan import valikosta. SoundCue on systeemi, jolla voi vaikuttaa yhteen tai useampaan ääneen, ja sen voi aktivoida Kismetin kautta. Siihen voi esimerkiksi yhdistää kolme eri ääntä ja sattumanvaraisesti soittaa jonkin niistä SoundCuen aktivoituessa Kismetissä. Projektin klikkausääniin ei tarvinut tehdä muuta kuin SoundCue, ja aktivoida se Kismetissä jonkin klikkaustoiminnon tapahtuessa.

Koska applikaatioon oli tärkeää saada yhtenäinen äänimatto, jottei koskaan tulisi täysin hiljaista hetkeä, ambient taustaäänien importoinnissa auto create sound cue kohdan lisäksi täytin include looping node kohdan. Tätä tarvitsin koska taustaäänet piti saada toistumaan niiden loputtua. include looping node kohta vaikuttaa SoundCueen lisäämällä sinne looping-noodin joka mahdollistaa äänien toiston niiden loputtua.

Kismetissä laukaisin ambient äänet Level Loaded -noodilla, joka aktivoi Play Sound -noodin, johon on laitettu halutun SoundCuen nimi.

6.11 Animaatio

Animaatioita voi tuoda Unreal Editoriin eksportoimalla 3D-ohjelmasta .FBX tiedoston. Editorissa voi myös tehdä animaatioita sen omalla animaatioeditorilla matineella. Matinee on Kismetin kautta aktivoitava noodi, jossa on matinee editorissa tehty animaatio. Editorissa pystyy tekemään animaatioita tai vaikka hallinnoimaan ja vaihtamaan aktiivista kameraa. Animointi editorissa toimii samoilla periaatteilla kun 3D-ohjelmissakin. sieltä löytyy samankaltaiset työkalutkin, kuten curve editori. Itse tein suurimman osan projektin animaatioista matineessa, mutta käytin myös blenderissä tehtyä animaatiota, jonka toin Unreal Editoriin .FBX tiedostona.



Unreal Matinee editori

Projektissa matineella animoituja asioita ovat kaikki laivan liikkuvat osat, kuten tykit ja keulan luukku. Myös idle-animaation ja kamera-ajon kamerat on animoitu matineella. Matineella actoreita animoitaessa on muistettava, että actorin täytyy olla jo oikealla ensimmäisellä keyframe paikallaan ennen matinee-noodin tekemistä. Tämä johtuu siitä, että kun matinee-noodia tehdään actorille, se luo automaattisesti sille myös ensimmäisen keyframen.

Tätä keyframea voi mennä muokkaamaan matineessa, jolloin se näyttää toimivan normaalisti, mutta matineen sulkeuduttua se palautuu taas omiin alkuperäisiin arvoihinsa joten animaatio ei toimi toivotusti.

Matineen lisäksi olen tehnyt animaation Unreal Editorin ulkopuolella laivan tykkeihin. Törmäsin projektini aikana ongelmaan tykkien koon kanssa. Tykit eivät alkuperäisessä koossaan mahtuneet sisään laivan rakenteisiin ja pienemmäksi skaalattuna ne eivät enää mahtuneet ampumaan ulos laivasta. Ratkaisin ongelman pienellä skaala-animaatiolla tykeissä, joka käynnistyy niiden animoituessa ulos tai sisään laivassa. Animaatio on tehty 3D-ohjelmassa morph target -animaationa ja sen jälkeen viety Unreal Editoriin .FBX tiedostona. Kun matineessa animoidaan .FBX tiedostona tuotua objektia, jossa on morph target animaatiota, siellä pitää luoda uusi morph weight raita, josta voi animoida objektin morph targetteja.

6.12 Game Icon

Game Icon -kuva näkyy esimerkiksi käynnistäessä applikaatiota start menussa. Kuvaa varten renderöin blenderistä laivan sivultapäin kuvattuna. Vein valmiin kuvan laivasta Photoshoppiin ja maalasin sen siellä mustaksi silhuetiksi, ja tein sille gradienteillä taustan, jossa myös lukee projektin nimi. Game Icon -tiedoston täytyy olla .ico muodossa. Toimiakseen Game Icon -kuvan täytyy olla pikseleissä yhtä korkea kuin se on leveäkin. Tallensin valmiin kuvan .png tiedostona ja konvertoin sen .ico tiedostoksi.

Kuvia voi konvertoida .ico tiedostoiksi esimerkiksi internetistä löytyvillä sivustoilla. Itse käytin www.icoconverter.com sivustoa. Kyseisellä sivustolla pyydetään käyttäjää tuomaan sivulle esmimerkiksi .png kuva. Sivustolla pyydetään myös valitsemaan halutut kuva koot .ico kuvalle. Koska .ico tiedostoa voidaan käyttää monessa eri paikassa se voi sisältää monta eri kokoa kuvasta. Konvertoinnin jälkeen valmiilla .ico kuvalla täytyy korvata Unreal Enginen oma GameIcon.ico-tiedosto, joka löytyy UDK:n asennuspaikasta Binaries/InstallData kansiosta.

6.13 Splash-animaatio

Splash-animaatio on animaatio, joka käynnistyy applikaation auetessa. Tyypillisesti splash-animaatio on tarkoitettu korostamaan käyttötuntumaa ja antamaan käyttäjälle ensimakua applikaatiosta.

Splash-animaation tein Blenderillä. Käytin animaatiossa samaa laivaa joka on käytössä itse lopputuloksessakin. Koska laivamalli on monessa eri osassa, animoinnin helpottamiseksi parentoin kaikki osat erilliseen kuutioon, jota sitten animoin, ja joka ei näkynyt renderöinnissä. Laivalle annoin mustan shadeless materiaalin, että sain vain laivan siluetin näkyville. Taustana on muokattu versio projektin GameIcon kuvasta, jota käytettiin tekstuurina shadeless materiaalissa.

Kun animaatiot oli tehty, renderöin splash-animaation ulos kuvasarjana ja yhdistin siihen äänet Blenderin videosekvenssieditorissa. Sieltä ulos renderöity lopullinen animaatio piti vielä muuttaa .bik tiedostoksi jotta UDK voisi lukea sitä. videoita voi konvertoida .bik tiedostoiksi

esimerkiksi RADvideo ohjelmalla. RADvideo työkalut saa ilmaiseksi ositteesta http://www.radgametools.com/bnkdown.htm. UDK etsii video tiedostojansa kansiosta /UDKGame/Movies, joten ne täytyy siirtää sinne .bik formaatissa. Splash video soi applikaation käynnistyessä Kismetin avulla. Kismetissä Level Loaded -noodilla käynnistetään Console Command -noodi, joka aktivoi MovieTest konsolikomennon, jolloin splash-animaatio käynnistyy heti scenen auetessa.

6.14 Flash käyttöliittymä

Seuraavaksi kerron hieman applikaation käyttöliittymästä ja sen tekemisestä. Vaikka kaikki projektin ominaisuudet ovat käytettävissä joko hiiren klikkauksilla tai näppäimistöllä, voivat jotkin käyttäjät kokea tavanomaisen käyttöliittymän intuitiivisemmaksi. Koska eri käyttäjät kokevat erilaiset asiat mukavemmiksi on tärkeää tarjota useampia eri tapoja hallita applikaatiota.

Tein applikaatioon käyttöliittymän antaakseni käyttäjille vaihtoehtoisen tavan päästä käsiksi laivan interaktiivisiin osiin ja muihin projektin ominaisuuksiin.
Ensimmäisenä testailin käyttöliittymän tekemistä interaktiivisena geometriana joka on itse scenessä ja liikkuu kameran liikkeiden mukana. Geometrialla tehdyssä käyttöliittymässä tuli ongelmia sen liikkeiden kanssa. Vaikka geometria kopioikin kameran liikkeitä kameran liikkuessa, se värisi häiritsevästi motion blur efektin tyylisesti. Koitin pitkään ratkaista kyseistä ongelmaa tuloksetta, joten päätin tehdä käyttöliittymän Flash ohjelmassa ja tuoda se .swf tiedostona Unreal Editoriin. Unreal Engine hyödyntää Scaleform ohjelmaa joka mahdollistaa Flash tiedostojen implementoinnin eri pelimoottoreihin. Scaleformia usein käytetään pelien käyttöliittymien toteutukseen.

Flash-tiedostoni on melko yksinkertainen. Se koostuu muutamasta nappulasta jotka lähettävät Fs-komennon aktivoitaessa ja kursorista joka seuraa hiiren liikkeitä. Nappuloille voi tehdä Fs-komennon flashissä actionina. Komennot täytyy nimetä järkevästi, koska niiden nimillä aktivoidaan Kismetissä noodeja, jotka kuuntelevat swf-tiedoston Fs-komentoja. Grafiikat ovat vektoreita, jotta ne skaalattuinakin näyttäisivät hyviltä.

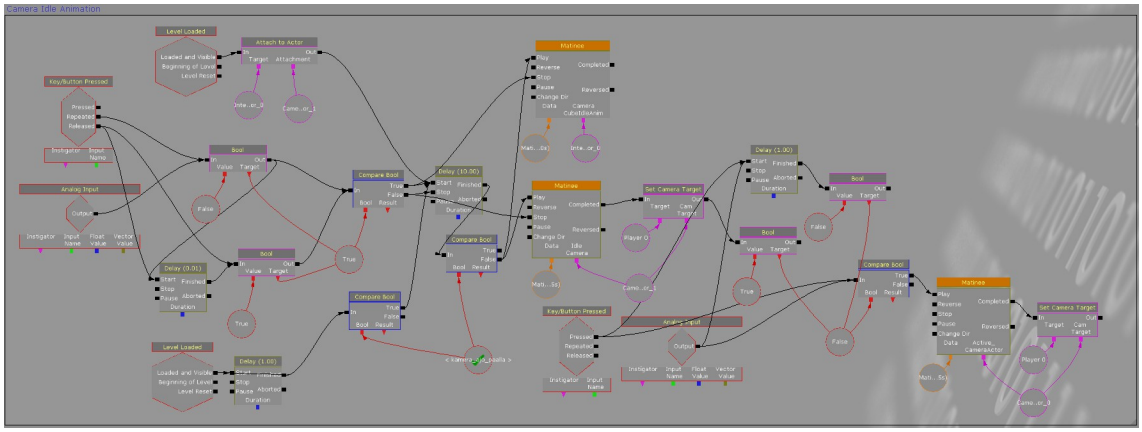
Flash-tiedoston voi importoida Unreal Editoriin .swf tiedostona ja se käynnistetään Kismetissä Open GFx Movie -noodilla. Noodiin määritellään pelaaja, jolle swf tiedosto käynnistyy, ja käynnistettävän swf tiedoston nimi. Swf tiedosto pystyy aktivoimaan tapahtumia Kismetissä käyttäen Fscommand käskyjä. Omassa tapauksessani swf tiedoston nappulat aktivoituessaan lähettävät oman nimensä mukaisen Fscommandin ja Kismetissä vastaavalla nimellä varustettu FsCommand-noodi aktivoituu.

6.15 Kamerat ja niiden käyttö

Applikaation scenessä on käytössä kolme eri kameraa. Yksi pääkamera, jolla pelaaja voi navigoida laivan ympärillä, yksi kamera idle animaatiolle, joka käynnistyy tietyn ajan kuluttua kun hiireen tai näppäimistöön ei ole koskettu, ja yksi laivan läpi kulkevalle kamera-ajolle. Aktiivista kameraa voi vaihtaa Kismetissä Set Camera Target -noodilla. Haluttaessa kameran vaihtumiselle voi tehdä animaation matinee-noodista löytyvällä director raidalla. Tässä tapauksessa Set Camera Target -noodi on kytkettävä matinee-noodin Completed kohtaan, jotta aktiivinen kamera vaihtuisi heti animaation loputtua.

6.15.1 Idle-animaatio

Applikaatiossa käynnistyy idle-animaatio, jos kontrolleihin ei ole koskettu kymmeneen sekuntiin. Idle-kamera toimii hieman samaan tapaan kuin päänavigointikamerakin. Se rotatoituu laivan keskellä olevan piilotetun kuution mukana, joka idle-animaation käynnistyessä alkaa pyörimään oman Z-akselinsa ympäri. Jokaisen käyttäjän teon jälkeen Kismet alkaa laskemaan Delay-noodilla kymmeneen sekunttiin. Jos käyttäjä koskee kontrolleihin, Delay-noodin lasku loppuu ja alkaa uudestaan. Kun aika on päässyt kulumaan, käynnistyy kaksi eri matinee animaatiota. Toinen matinee animaatio animoi director trackillä kameran vaihtumisen idle animaatiossa käytettävään kameraan, ja toinen aloittaa ilde animaation kameraa ohjaavan kuution loopilla pyörivän animaation. Jos käyttäjä koskee kontrolleihin idle-animaation aikana, aktivoituu matinee-animaatio, jossa kamera vaihdetaan takaisin päänavigointikameraksi ja pyörimisanimaation matinee-noodia käsketään lopettamaan animaatio.

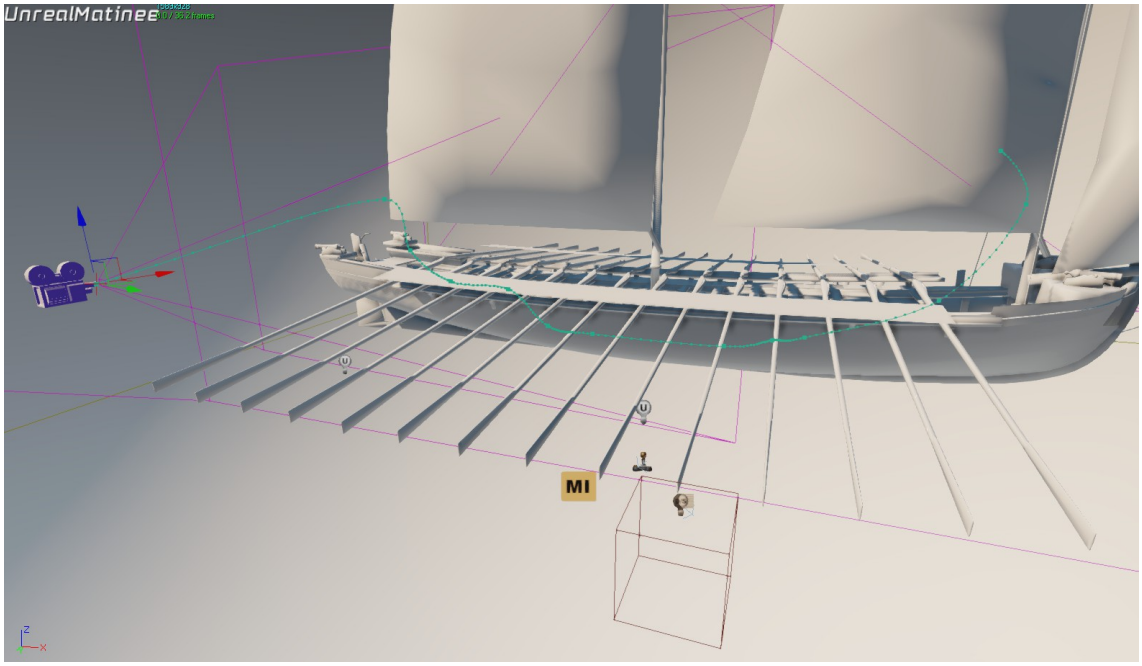


Idle-animaatio Kismetissä

Ennen idle-animaation käynnistymistä Kismet myös tarkistaa Compare Boolean -noodilla onko applikaation kamera-ajo käynnissä. Jotta kamera-ajo ei keskeytyisi idle-animaatiolla, Kismet-systeemi keskeyttää idle-animaation käynnistymisen kamera-ajon ollessa päällä.

6.15.2 Kamera-ajo

Tein applikaatioon laivan läpi kulkevan kamera-ajon, jotta käyttäjät pääsisivät lähemmin tarkastelemaan laivan sisäosia. Käyttäjän aktivoidessa kamera-ajon aktiivinen kamera vaihtuu kamera-ajossa käytettäväksi animoiduksi kameraksi. Aktiivisen kameran vaihtaminen tapahtuu Kismetissä samalla tavalla kuin idleanimaatiossakin. Aktiivisen kameran vaihtamisen lisäksi Kismetissä aktivoituu matinee-noodi, jolla on animoitu itse kameraa joka liikkuu laivan läpi. Kamera-ajon alettua Kismetissä myös muutetaan boolean-noodia joka kertoo idle-animaatiolle että kamera-ajo on päällä ja estää idle-animaation käynnistymisen. Itse kamera-ajoa animoidessa oli jatkuvasti tarkistettava ettei kamera liukuisi minkään geometrian läpi laivan ahtaissa siästiloissa. Tätä myös vaikeutti tykkien animaatiot ja se, että ne pystyvät olemaan kahdessa eri paikassa laivassa.



Kamera-ajon polku Unreal Editorissa

Kameroiden animoimisen helpottamiseksi kannattaa animoidessa tarkastella mitä kamera näkee. Unreal Editorin perspektiivi-ikkunan pystyy vaihtamaan halutun kameran näkymäksi avaamalla sen matinee-animaation ja painamalla sen kameraryhmän vieressä olevaa pientä kameran kuvaa. Näin näet jatkuvasti mitä kamera näkee, eikä animaatiota tarvitse testata itse applikaation kautta.

6.16 Navigoinnin ohjevideo

Halusin sisällyttää projektiin jonkinlaisen ohjeen, miten kontrolleita käytetään, jotta käyttäjien ei itse tarvitsisi hakea ja keksiä niitä. Ensin ajattelin toteuttaa sen tekstinä, jota olisi tukemassa muutama eri kuva esimerkiksi itse hiiren näppäimistä. Nähtyäni kuitenkin MiracleAircraft applikaation ohjevideon, päätin tehdä hieman samanlaisen.

Tarkoituksena oli tehdä video, jossa näytettäisiin samaan aikaan applikaation ruutua, ja sen videon päällä toinen pienempi video, jolla näkyisi samaan aikaan tehdyt liikkeet ja napin painallukset hiirellä. Kuvasin siis samanaikaisesti kahta eri videota, jotka halusin myöhemmin yhdistää toisiinsa jossain videonmuokkausohjelmassa.

Nauhoitin Camtasia-ohjelmalla videota suoraan applikaatiosta, ja samaan aikaan oman käteni ja hiireni tapahtumia toisella kameralla. Tarkoitus oli käydä ruudulla läpi applikaation kaikki navigaatio-ominaisuudet, ja samalla näyttää tärkeimpiä ominaisuuksia. Huomasin hiirtä kuvatessani että kaikki käteni liikkeet ja nappuloiden painamiset piti olla melko suurieleisiä, jotta ne ymmärtäisi videosta.

Yhdistettyäni videot After Effects -ohjelmassa toin lopputuloksen Flashiin ja julkaisin Unreal Editorissa käytettävän .swf tiedoston. Törmäsin kuitenkin uuden .swf tieoston kanssa ongelmaan. Aina käynnistäessä applikaatiota uuden flash menun kanssa editori kaatui heti käynnistyessään. Pitkän tutkimisen jälkeen törmäsin UDK foorumeilla keskusteluun jossa puhuttiin UDK:n lisensoinnista ja ilmaislisenssin rajoituksista. Keskustelusta tuli ilmi että ilmaislisenssillä Scaleform ei anna Flash tiedostojen pyörittää videotiedostoja. Ratkaisin ongelman käynnistämällä videon Unreal moottorista eikä flash tiedoston sisältä. Video käynnistyy siis nyt ohjenappulan klikkauksella lähtevästä Fscommand-käskystä, joka Kismetissä aktivoi movietest komennon Console Command -noodilla.

7 johtopäätökset

Koska projekti on tarkoitettu pöytäkoneilta käytettäväksi, olisi ollut hyvä sisällyttää siihen jonkinlainen grafiikoiden säätömahdollisuus. Käyttöliittymään olisi ollut hyvä saada nappi, josta olisi päässyt grafiikka-asetuksiin käsiksi. Grafiikka-asetuksissa olisi

vähintään muutamia erilaisia resoluutiovaihtoehtoja, ettei käyttäjän tarvitsisi vaihtaa niitä konsolikomennon kautta.

Olisin mielelläni halunnut tehdä projektista myös tablettiversion. Vaikka projekti olisikin soveltunut hyvin tabletille, sen toteuttaminen olisi vaatinut liikaa aikaa. Tablettiversioon olisi ollut mukava sisällyttää uusia ominaisuuksia, kuten tabletin gyroskoopin. Gyroskooppia olisi voinut käyttää vaikka ohjelmassa navigoimiseen.

Lopputuloksessa menin hieman liian pitkälle laivamallin optimoinnissa. Malli on mielestäni hieman liian yksinkertainen, ja teksturoinninkin kanssa olisi voinut viettää enemmän aikaa. Laivamallille olisi ollut toivottavaa tehdä myös jonkinlainen ympäristö. Laiva olisi voinut olla esimerkiksi saaristoisessa meressä tai vaikka kuivatelakalla.

Projektiin olisi voinut vielä sisällyttää muitakin kamera-ajoja yhden laivan läpi kulkevan lisäksi. Aktiivisia kamerakulmiakin olisi voinut olla lisää. Yksi mahdollinen paikka olisi ollut kannen päällä oleva aktiivinen kamerakulma, josta olisi voinut hiirtä käyttämällä katsella ympärilleen. Olisin myös voinut animoida useampia asioita laivassa. Purjeet olisivat voineet liikkua hieman tuulessa ja airot olisivat voineet liikkua hiiren klikkauksesta tai käyttöliittymästä aktivoituna.

Vaikka olisin halnnut lisätä paljonkin erilaisia ominaisuuksia ja viilauksia projektiini, joita en saanut opinnäytetyöni rajoissa tehtyä, olen lopputulokseen silti hyvin tyytyväinen. Sain kuitenkin kaikki tärkeimmät ominaisuudet tehtyä jotka alunperin halusinkin projektiini toteuttaa. Opin paljon Unreal Enginestä ja yleisesti reaaliaikamoottorien kanssa työskentelystä joten sekin tavoite työssäni toteutui.

8 Lähteet

Ernest Adams 2009. The Designer's Notebook: Sorting Out the Genre Muddle. Gamasutra. [verkkosivu] <http://www.gamasutra.com/view/feature/132463/the_designers_notebook_sorting_.php?page=4> (luettu 25.4.2014)

Corinth 2013a. App store. [verkkosivu] <https://itunes.apple.com/fi/app/ancient-egypt-virtual-3d-interactive/id537986427?l=fi&mt=8> (luettu 22.4.2014)

Corinth 2013b. App store. [verkkosivu] <https://itunes.apple.com/us/app/miracle-aircraft-3d-reconstruction/id551357352?mt=8> (luettu 22.4.2014)

Epic Games, Inc 2011. Creating a mouse interface. UDN. [verkkosivu] <http://udn.epicgames.com/Three/DevelopmentKitGemsCreatingAMouseInterface.html> (luettu 22.4.2014)

KdgDev 2011. Game Development Stack Exchange. [verkkosivu] <http://gamedev.stackexchange.com/questions/12709/move-the-location-of-the-xyz-pivot-point-on-a-mesh-in-udk> (luettu 22.4.2014)

Mcampagnini, Richisbored 2012. Google code. [verkkosivu] <http://code.google.com/p/ase-export-vmc/> (luettu 22.4.2014)

NATO_chrisjm 2012. EXTERNAL PACKAGES, LEVEL PACKAGE; AND SAVING ASSETS. 3D Buzz [verkkosivu] <http://www.3dbuzz.com/forum/threads/192989-External-Packages-Level-Package-and-saving-assets> (luettu 22.4.2014)

Suomenlinna. Tykkisluuppi. [verkkosivu] <http://www.suomenlinna.fi/matkailijan_suomenlinna/kayntikohteet/tykkisluuppi> (luettu 22.4.2014)

ThatDutchGuyUDK 2013. Tutorial 1: Move and rotate objects in UDK using Kismet. Youtube. [verkkosivu] <http://www.youtube.com/watch?v=KkXFyyz4emM> (luettu 22.4.2014)

Tykkisuuppi applikaation asennustiedosto

Alla olevasta linkistä voi ladata Tykkisluuppi applikaation asennustiedoston.

(https://www.dropbox.com/sh/y0j21284aeu94cl/y0y55ky9cL)

Asennusohje: tuplaklikkaa UDKinstall-Tykkisluuppi.exe asennustiedostoa. Hyväksy asennustiedoston käyttöehdot. Valitse paikka mihin applikaatio asennetaan ja aloita asennus. Applikaation voi käynnistää kun asennus on valmis. Käynnistystiedosto löytyy applikaation asennuspaikasta Binaries/Win32/ kansiosta nimellä UDK.exe

Kismet kaaviot suurempina kuvina

(https://www.dropbox.com/sh/3jbatuypk6empok/PrZaXFdBdP)